THE IMPACT OF NEW COMPUTER TECHNOLOGY ON ACCELERATOR CONTROL

E. Theil
V. Jacobson
V. Paxson

Real Time Systems Department
Lawrence Berkeley Laboratory
University of California
Berkeley, California 94720

Abstract

This paper[1] describes some recent developments in computing and stresses their application to accelerator control systems. Among the advances that promise to have a significant impact are i) low cost scientific workstations; ii) the use of "windows", pointing devices and menus in a multitasking operating system; iii) high resolution large-screen graphics monitors; iv) new kinds of high bandwidth local area networks. The relevant features are related to a general accelerator control system. For example, we examine the implications of a computing environment which permits and encourages graphical manipulation of system components, rather than traditional access through the writing of programs or "canned" access via touch panels.

Introduction

Trends in computer hardware and programming methodology are moving too fast for any one paper to present a comprehensive survey of ideas that might affect accelerator control systems. Nevertheless, there are several clearly identifiable developments which started within the fairly recent past (3–4 years ago), are continuing today, and which hold considerable promise for accelerator control in the near future. In some cases, early versions of these tools and ideas are already appearing in existing control systems [Shering]; others are in the process of development, or in the design stage.

This paper, then, is not a survey, but a discussion of a few of the most significant of these ideas, together with some examples of what is possible when they are applied to complex accelerator operation. The viewpoint is that of the user, rather than the engineer; i.e., we are concerned with *what* we should do with these new tools, rather than *how* to do it. Additional examples are described in [2] and [3], which may be considered as companion papers to this one.

One area that has already been written about extensively [4,5] is the use of microprocessors in a distributed system, particularly at the "front end", for device control. Consequently, we will not discuss that development here. Our emphasis will be on commercial hardware and software trends, particularly as they can affect the interaction between operator and accelerator.

There are three major, highly interdependent areas. They are:

- The introduction of high performance, low cost workstations.
- Emerging network standards and network transparency. The frequently heard description of computer networks as comprising a modern Tower of Babble has been unfortunately accurate, but this situation is showing signs of considerable improvement.
- New software developments with respect to the human interface. Workstations incorporate many features that are a result of both software and human factors research. Among these are pointing devices and screen windowing systems. Less obvious, but ultimately of great importance for control systems, is the direct manipulation of representations of physical objects and the associated technique of "visual programming."

The challenge to the designers and builders of accelerator control systems is to use these powerful new developments in an appropriate and effective manner. As a first step in this process, we can try to provide some examples of how they might be applied.

Workstations

These devices are now commonplace, with MicroVAX, SUN and Apollo computers being representative systems. They all feature a more or less powerful CPU, high resolution CRT output device and run familiar, de facto standard operating systems, the most common of which is a version of Unix.[2] Finally, their cost is often a factor of five to ten less than their minicomputer ancestors. These systems provide a high level of programming support. That is, a number of tools are supplied to enhance the programming environment and make communication with peripherals and the user a relatively straightforward procedure.

An important part of the programming and user interface is provided by a window system and a standard graphics package, such as GKS, with a mouse for a pointing device. Another major element of the system is the *integrated* network hardware and software. These workstations are typically designed to be part of a network of computers. This basic philosophy, combined with advances in software for distributed processing and emerging network standards (both discussed below) means that it is simpler and cheaper to incorporate these machines into a distributed control system than it has been in the past, in terms of programming and network interfacing. The implications of these facts bear further examination. Let us suggest just a few.

First, simply in terms of power and cost, these systems present new opportunities for accelerator control. The power available should not be underestimated: one can buy a 4 MIPS workstation (comparable to a Vax 8600 at 15% of the Vax cost) as a modeling number cruncher or database machine. Because workstations are inexpensive and easy to network, one is led to consider the notion of a control system distributed not just at the "front end" (device level), but at the "back end" (operator level) as well. For example, a simple way to increase system performance would be to dedicate one or more workstations solely to the system database. Other stations could be dedicated to subsystems (e.g., vacuum or beamline monitoring and control), much as is done today, but with much more power and greater ability to display information. Since each station provides its own graphics capabilities, the systems avoid the usual bottleneck of one or two central computers competing for display screens.

In addition, system down time can be minimized, because spare machines and consoles are now economically practical. "Spares" can, in fact, be used for off-line analysis or off-line hardware checkout, as well as program development, without impacting operations. Physicists can monitor operations or examine data in their offices, using workstations that tie into the control system's database.

These are advantages that are primarily a function of cost. For at least some of them, one could substitute home grown single-board microprocessors or inexpensive personal computers and, superficially at least, gain some of the same advantages. However, the introduction of workstations into the control system can have a more profound impact on both systems and applications programming.

[1] Work supported in part by the U.S. Department of Energy under Contract Number DE-AC03-76SF00098

[2] It is worth noting that progress toward a standard for *real time* Unix is under way, with support from major corporations.

CH2387-9/87/0000-0529 $1.00 © IEEE

First, considerably less effort has to go into traditional systems programming. Many of the usual hurdles — construction of network software and low level graphics interfacing, for example — are avoided by the fact that network and graphics software are integrated into the system. Second, application programming can be carried out at a higher and more productive level because the systems come with powerful debugging tools and source control aids. That, combined with the graphics interface and screen windows, tends to lead programmers and users into new ground. Control tasks become more graphics-oriented and interactive [2] (see Fig. 1). Moreover, not only is the output more graphic, but *input* is as well. The ability to select menus, move icons, create and manipulate sliders, knobs and dials in any of several windows on the screen is a natural counterpart of the way in which operators interact with an accelerator. This approach to the operator interface can be significantly faster, simpler and more effective than the traditional styles that combine touchpanels and keyboards. We will return to this important point later.

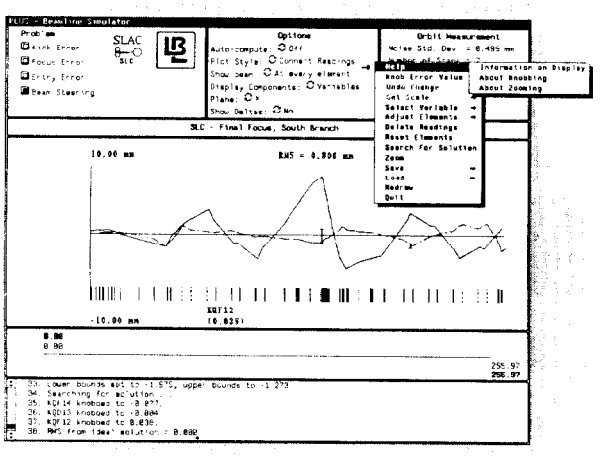

Figure 1: A high-level control task with graphics interface

Thus, we feel the truly significant implications for accelerator control are intimately tied to the graphics capabilities of the workstation, together with emerging standards in communications and software. For this reason, we discuss those topics first, and then return to new ways of interacting with control programs.

Network Standards and Transparency

We give two important examples of emerging standards: MAP as a network hardware based standard and NFS as a software based counterpart.

MAP — General Motor's Manufacturing Automation Protocol — is based on the first few layers of the OSI seven layer model. Currently, it exists as a 10Mbps broadband token bus, with baseband versions under development. Since it is a token bus, it has the advantage that the maximum wait time before a node can transmit is well determined, in contrast to a collision avoidance broadcast protocol, typified by Ethernet.

It is, however, the broader implications of the standard and its industrial sponsorship that are particularly important in this context. One can buy off-the-shelf MAP board-level interfaces, similar to the types available for Ethernet, along with hardware that permits inter-networking between, say, MAP and Ethernet, or MAP and Proway. Chip sets and board level bus controllers for MAP have been available since late '85. By 1988, MAP products will be fully integrated into many workstations: that is, one will be able to plug a MAP cable into a workstation already equipped with MAP communication handlers.

This is not a brief for MAP, just an example. The point is not that MAP is a solution to the control network problem, but that these emerging standards and commercial support mean that control systems designers and implementors can avoid the drain of resources that have been associated with networks in the past. Also, one can design a network now using, say Ethernet, and know that the software and hardware investment is safeguarded. At the appropriate time, changing to MAP or another standard will require changing at most a board per computer.

The second example deals with the problem of shared data in distributed systems. This difficult problem is also being addressed by an emerging standard, this one in software. The development is exemplified by NFS — Network File System, which is available from several workstation vendors now. NFS has a number of important features for distributed control. For example:

- Transparent remote file access anywhere on a LAN. One simply mounts the appropriate part of a remote server's file system as a directory on the client. Thereafter, no new calls or commands are required; that file directory is available on the local machine, just as if the remote disk were mounted locally.
- High performance: access times for remote files are about 90% of local files.
- NFS supports the ability to make REX (remote execution) calls for compute-intensive tasks on remote systems.
- NFS supports file and record locking during concurrent access.

As in the case of MAP, our interest here in NFS is not because of its technical merits *per se* but rather, because it is yet more evidence of the way in which this problem, which is of direct concern in the design of control systems, is being addressed by industry.

Windows

Among the developments centered around the user interface, the ones that seem particularly appropriate for accelerator control are the use of a pointing device (usually a mouse), menus (in various pop-up styles) and window systems [1].

An accelerator may be thought of as a collection of tightly coupled subsystems: lattice, vacuum, particle beam, etc. Changes to any one subsystem can and frequently do have significant effects on the others. One of the advantages of a window system in operator consoles is that it provides the ability to monitor and control several processes or subsystems on the same screen at the same time. Roughly speaking, a process is attached to a single window, but because the operating system is multi-tasking, the processes can execute concurrently in their windows. Processes are able to communicate with each other at high band-widths. In this way, the window system presents a visual model of parts of the accelerator. Combined with the operator's ability to create, remove, resize and rearrange new windows as circumstances dictate, this development is highly germane to accelerator control. An example is shown in Fig. 2.

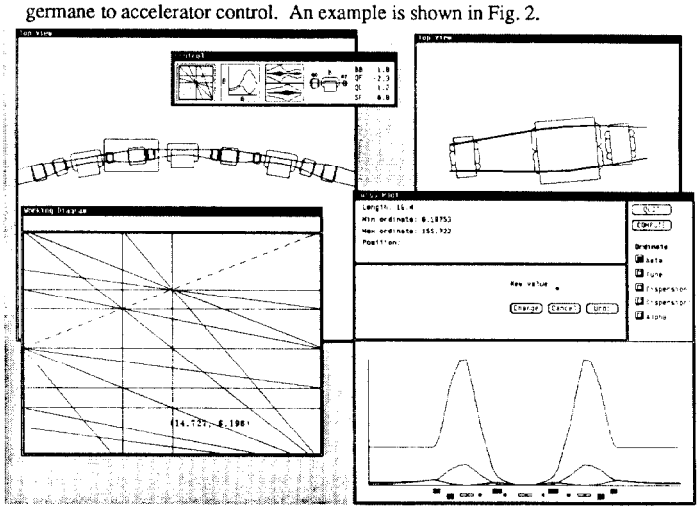

Figure 2: Prototype control tasks in a window system

As with most new developments, care should be exercised: studies [6] suggest that the manipulation of windows may interfere with getting the job done. Fortunately, window placement can be defaulted and/or linked together.

Direct Manipulation and Visual Programming

Several kinds of models may be involved in the windowing process. One obvious example is the use of mathematical models that describe the interaction of the beam with the machine lattice and insertion devices [3] A simpler model might drive a display that represents a beamline and the status and strength of each of its elements [2]. In either case, a highly graphic interface, combined with support for the operator to simply point to objects in order to change their values or status, leads to the notion of *direct manipulation.*

To illustrate the concept, imagine a system that allows students to bend the curves of a polynomial function displayed on the screen and watch the coefficients or the derivative function change. The same idea can be applied to accelerator control with several happy consequences. The functionality of the displays tends to be intuitive, and one can use the underlying models to learn about the accelerator by simulation. The idea is not limited to high-end workstations; Fig. 3 illustrates direct manipulation of laboratory instruments using a package called LabVIEW and the Macintosh computer. Clearly, the display has the virtue of intuitive understanding.

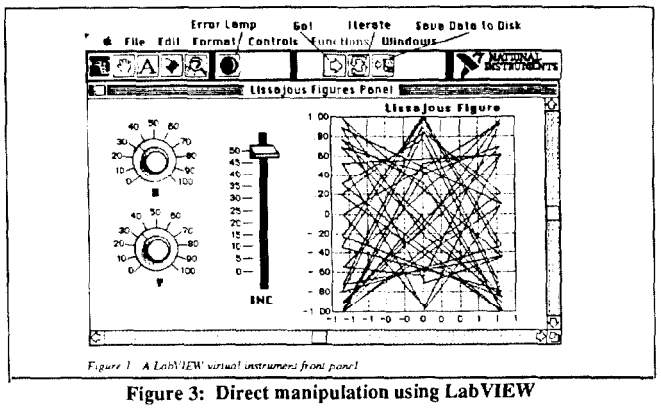

Figure 3: Direct manipulation using LabVIEW

Fig. 4 represents another example, abstracted from our own work. It displays a set of beamlines, each with its complement of magnets. Rather than type a series of commands, an operator can select any component just by clicking the mouse on it. The magnet's colloquial name is then displayed, together with its status (on/off, say), and strength. By pressing a mouse button, a menu presents several other options. Using the menu, the entire beamline of which that magnet is a part can be selected automatically. Now, the beamline status itself can be changed; it becomes the object being manipulated. (In order to display and control complicated objects like beamlines in a general way, we have developed a simple "picture language" [2]).

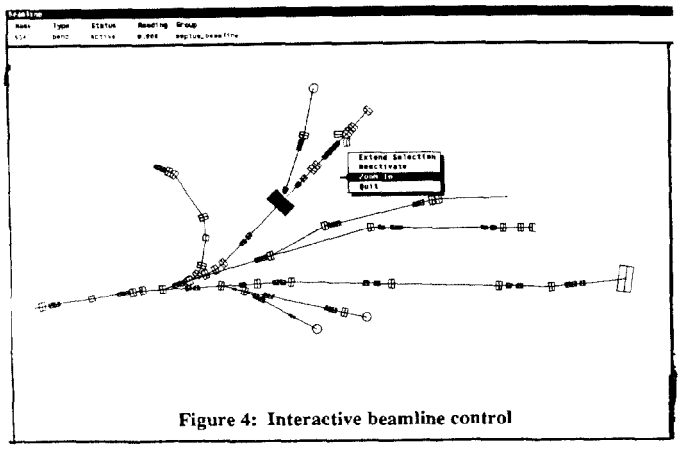

Figure 4: Interactive beamline control

Direct manipulation avoids the traditionally troublesome problem of specifying objects by name. Rather than needing to know the names of any of perhaps hundreds or thousands of similar objects, the user need only specify *that* device by pointing and clicking, in order to control it.

An idea related to direct manipulation is *visual programming.* If one manipulates objects directly on the screen, it should be and is possible to automatically log the sequence of internal operations that result in the desired actions. That sequence can be saved and becomes a procedure or program that can be recalled and executed rapidly on demand. An example is the capture of a Postscript procedure that reproduces a MacPaint illustration. In addition to automatically generating needed programming procedures by graphically describing the outcome, another important consequence is that the procedure, not the data, is stored and shipped. Since typically the former is much smaller than the bit-mapped display, the result is that traffic over the control network can be greatly decreased.

Summary

- Emerging standards in operating systems, graphics and networks create an opportunity to develop control software with an interface that is highly interactive and which itself forms a conceptual model of parts of the accelerator. Such software is being built today.
- "Buy, don't build" is more true today than ever before. It is possible to buy well integrated computer hardware from a variety of vendors and avoid much of the traditional low level interfacing and software work associated with distributed systems. Generally, one also gets a large software base, including programming tools.
- As the discussion of MAP indicates, using off-the-shelf also guards against obsolescence. Plug compatible upgrades of hardware with standard protocols and interfaces significantly increases the life of software, thereby decreasing maintenance costs.
- As a corollary, valuable human resources are available for greater effort in applications software. The tailoring of a control system for a specific facility can then emphasize the needs of the various types of users (machine physicists, operators, engineers), rather than the constraints imposed by the system itself.

References

[1] G.Shering, "Consoles and Displays for Accelerator Operation," in Computing in Accelerator Design and Operation, Busse & Zelazny, eds., Springer-Verlag, 1983, pp. 481–487

[2] V. Paxson, V. Jacobson, E. Theil, "A Scientific Work-station Operator-Interface for Accelerator Control," presented at this IEEE Conference, 1987

[3] M. Lee, et al., "Modern Approaches to Accelerator Simulation and On-line Control," presented at this IEEE Conference, 1987

[4] J.Alter, et al., "Replacing Mini-Computers by Multi-Microprocessors for the Lep Control System," IEEE Trans. on Nucl. Sci., NS-30, No. 4, August, 1983, pp2287–2289

[5] S. Magyary, H. Lancaster, et al., "Operating Experience with a New Accelerator Control System Based Upon Microprocessors," IEEE Trans. Nucl. Sci., Vol. NS-28, NO. 3, June, 1981, pp2201–2203

[6] B. Shneiderman, Designing the User Interface, Addison-Wesley, 1987

[7] T. Clifford, et al., "Experience in Using Work Stations as Hosts in an Accelerator Control Environment", presented at this IEEE Conference,1987